AMD
Smarter Choice

Gettin' Procedural

Jeremy Shopf
3D Application Research Group

Adding Procedural Content (30 mins)

AMD Smarter Choice

Basics

- Why should I care?
- Noise, etc.

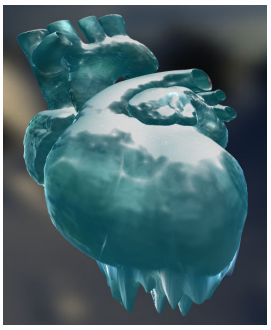
Procedural Surfaces

- Layered ice

Procedural Geometry

- Growing icicles on the GPU

Demo



Why should I care?

AMD
Smarter Choice

Low storage space

Resolution-independent

Flexibility

- Change parameters dynamically
- Based on time, specific object, events, etc.

Variation

- The number of art assets is (typically) increasing
- Parameterizing the appearance of an object procedurally allows identical objects with identical shaders and art assets to appear much different

Noise

AMD
Smarter Choice

What is noise?

Noise adds pseudo-randomness

Properties of Perlin Noise

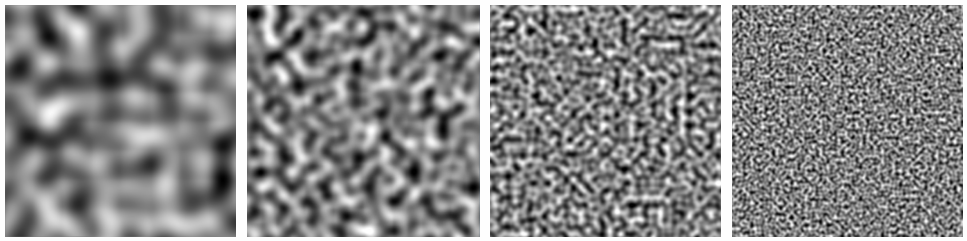
- Smooth
- Correlated (it’s going to look the same next time)
- Band-limited (values are within a certain range)

Mixing it up a little

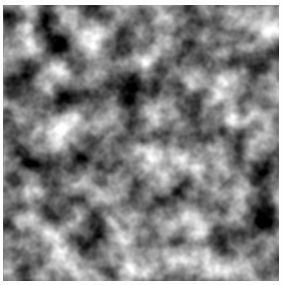
AMD Smarter Choice

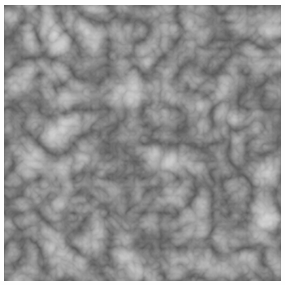
F = 2 F=4 F=8 F=16

Noise



*1/2 + *1/4 + * 1/8 + *1/16 =

Abs() =



Summed octaves **Turbulence**

See Natalya Tatarchuk's slides from Monday at [http://ati.amd.com/developer/techpapers.html]

Google Ken Perlin's talk "Making Noise" from GDC99

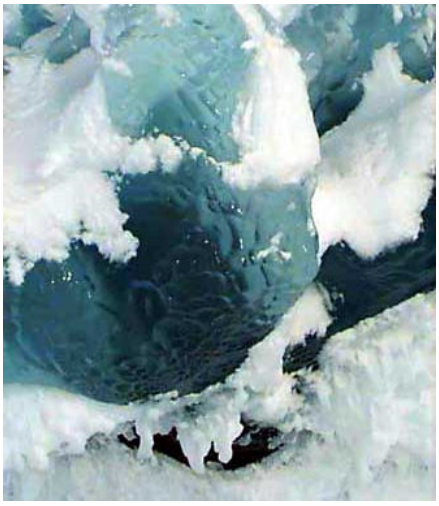
Procedural Surfaces

AMD Smarter Choice

Making advanced rendering techniques procedural

Goal: Ice!

http://www.leatherwoodonline.com



Ingrid McGaughey

Getting Started

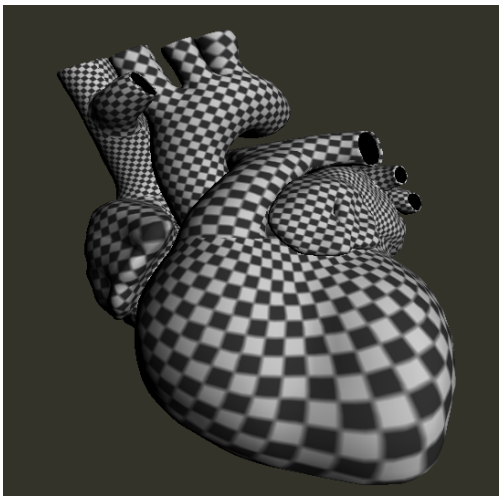
AMD
Smarter Choice

The secret of making procedural shaders is thinking of appearance in layers and masks

Try and match features in the noise to appearance

Apply math to shape these features to do your bidding

We will examine a few such cases shortly…



Ice color layer

AMD
Smarter Choice

Thick ice is typically not a uniform color

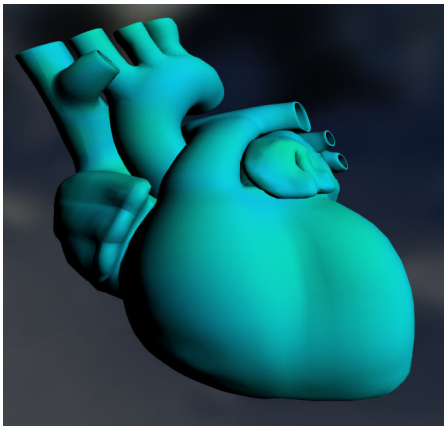
- Some blend of white, blue and green

Lerp between two or more colors based on noise!

Make color based on thickness

- Scattering properties of ice attenuates some wavelengths more than others

Make thickness based on noise

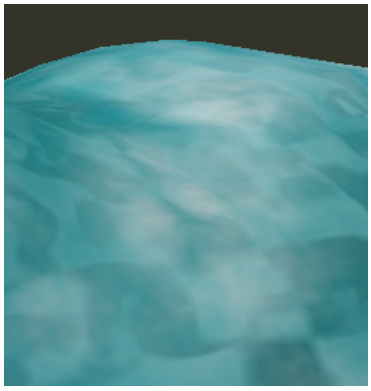


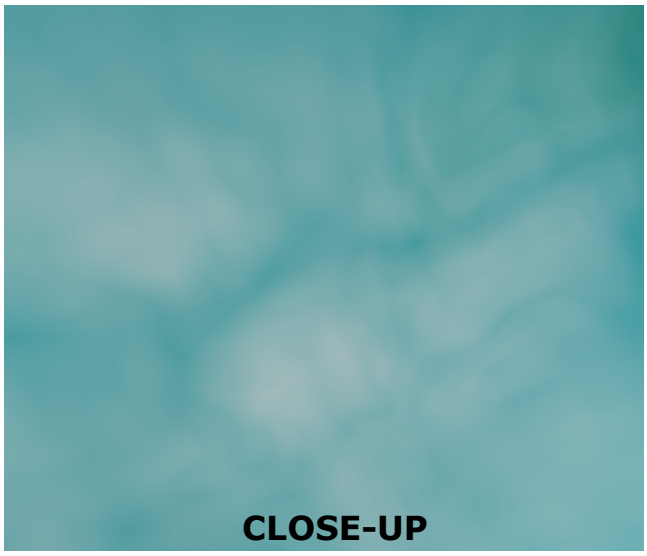
Ice cloudy layer

AMD
Smarter Choice

Cloudy value is turbulent noise

Lerp cloudy color with ice color that was determined earlier





Adding parallax

AMD
Smarter Choice

Ice consists of many physical layers

- Typically can be divided into 2 or 3 important ones

Oat's "Rendering Gooey Materials with Multiple Layers" technique [http://ati.amd.com/developer/techreports.html]

- Deals with parallax in any kind of layered material
- (Not just gooey ones!)

Oat's Multi-layered Approach

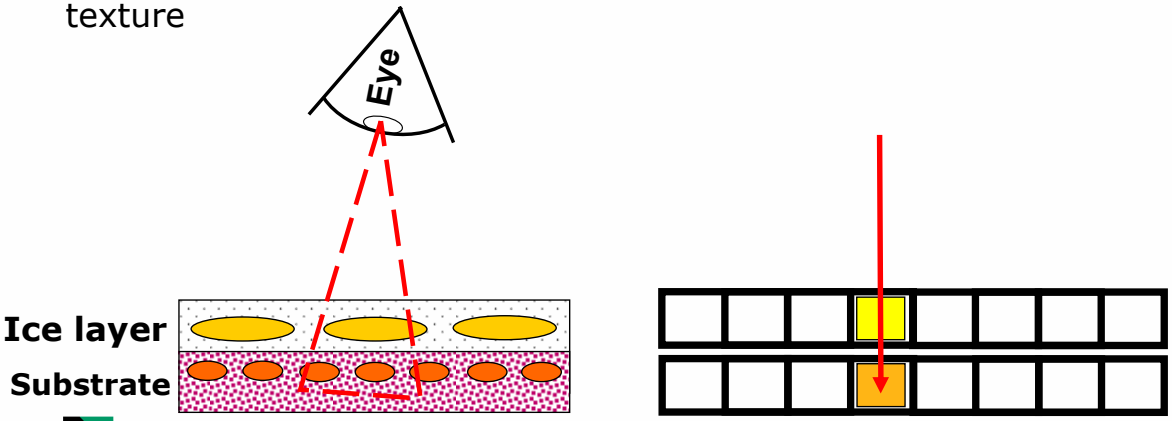
AMD
Smarter Choice

Make the material look volumetric

Depth parallax

- **Shift in apparent position due to change in view**
- Inner layer shifts with respect to outer layer
- Shift is more pronounced as depth increases

Can't use surface layer's UV coordinate to sample inner layer's texture



Oat's Multi-layered Approach

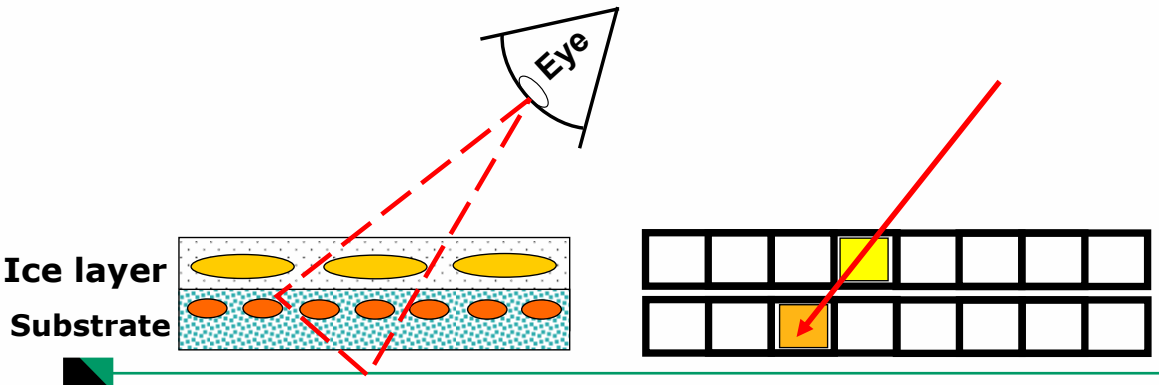
AMD Smarter Choice

Make the material look volumetric

Depth parallax

- Shift in apparent position due to change in view
- **Inner layer shifts with respect to outer layer**
- Shift is more pronounced as depth increases

Can't use surface layer's UV coordinate to sample inner layer's texture



Oat's Multi-layered Approach

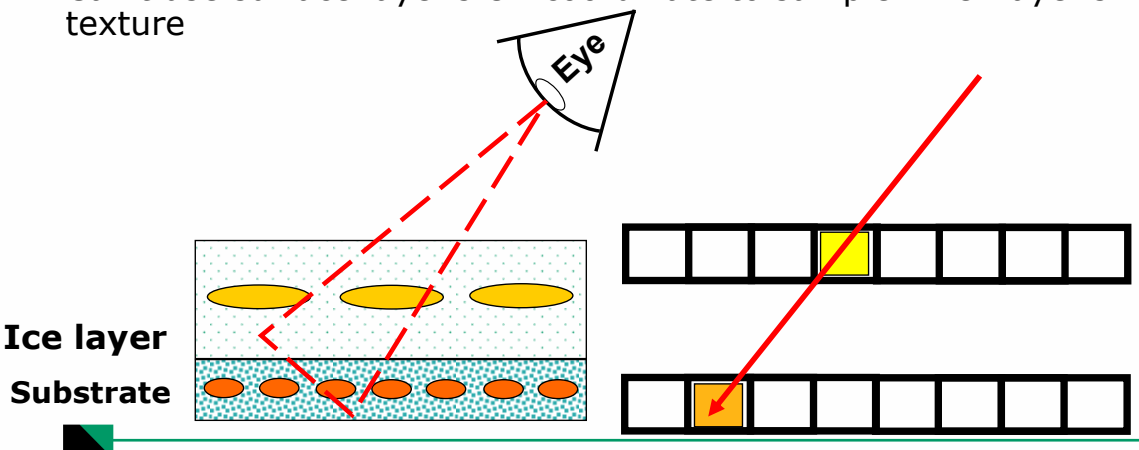
AMD Smarter Choice

Make the material look volumetric

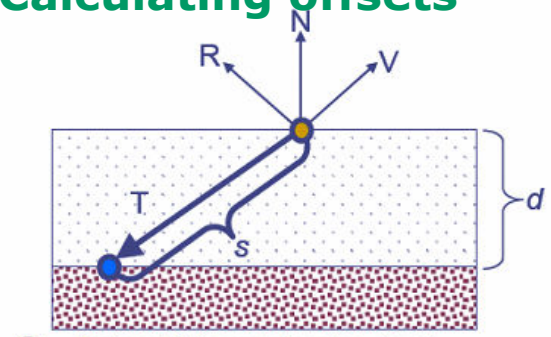
Depth parallax

- Shift in apparent position due to change in view
- Inner layer shifts with respect to outer layer
- **Shift is more pronounced as depth increases**

Can't use surface layer's UV coordinate to sample inner layer's texture



Calculating offsets



= Outer UV coordinate: <u,v>

= Inner UV coordinate: <u',v'>

$$\vec{R} = -\vec{V} - 2 * dot(-\vec{V}, \vec{N}) * \vec{N}$$

$$\vec{T} = \langle \vec{R}_x, \vec{R}_y, -\vec{R}_z \rangle$$

$$s = d / \left| \vec{T}_z \right|$$

$$\langle u', v' \rangle = \langle u, v \rangle + s \langle \vec{T}_x, \vec{T}_y \rangle$$

- Layers are assumed to be parallel in tangent space
  - Layer depth **d** is homogeneous for a given layer
- Find inner layer's texture coordinate
  1. Find view vector = **V**
  2. Reflect **V** about **N**ormal (from normal map) = **R**
  3. Reflect **R** about surface plane = transmission vector **T**
     - In tangent space, we simply negate **R.z** component
  4. Find distance **s** along **T** to inner layer: Function of distance **d** between layers
  5. Use **T** and **s** this to find inner layer's texture coordinate

Snow

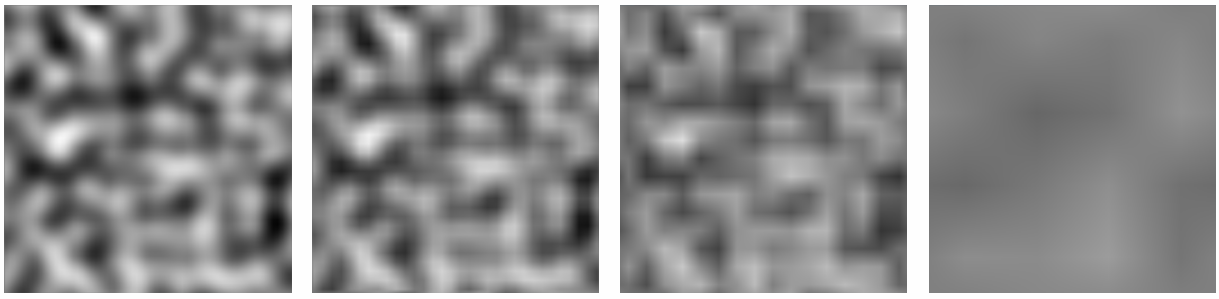
AMD Smarter Choice

Add snow dusting on surfaces oriented towards the sky
- Snow amount = N dot "Up"

Make snow accumulation boundary uneven
- 1st intuition: Mask N dot "Up" by noise

Doesn't work so well because you only want mask out regions at the edge of the snow

Solution: Index noise texture with mip bias based on N dot L



Level 0 **Level 4** **Level 6** **Level 7**

Snow (cont.)

AMD
Smarter Choice

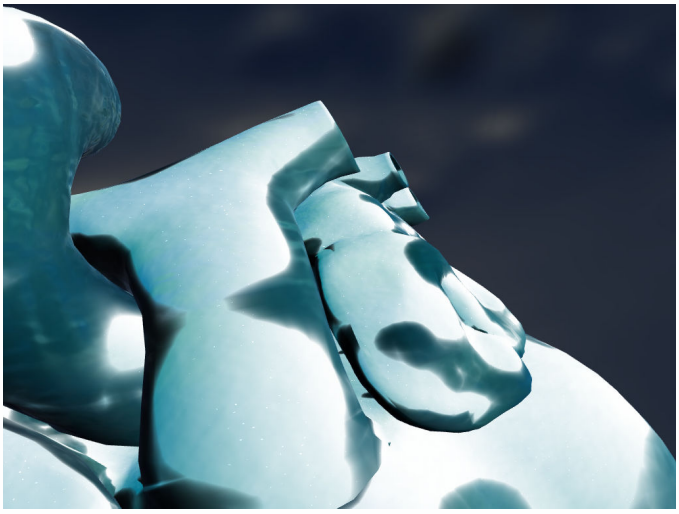
Just N.y

Snow (cont.)

Just N.y * Hi-Freq Noise

Snow (cont.)

AMD
Smarter Choice



Just N.y * Low-Freq Noise

Snow (cont.)

AMD
Smarter Choice

Just N.y * Mip-biased noise (half-chain)

Snow (cont.)

AMD
Smarter Choice

Just N.y * Mip-biased Noise

Snow Lighting

AMD
Smarter Choice

Snow bump is generated procedurally from summed octaves of noise

- Add snow bump to geometric normal to get smoothly changing lighting
- Lerp between ice color and snow color to get shading that mimics effects of scattering in snow and ice :

```
lerp( cIceColor, cSnowColor, fNdotL )
```

Snow Sparkle

AMD Smarter Choice

Snow sparkle is harder than you think

- 1st Intuition: calculate specular using random noise vector as normal
- Requires uniform uv mapping
- Very high res, high frequency noise texture

Decent solution: Use 3D position to index 3D noise function, add the view vector, use frac function to further randomize things

Sparkle:

```
float specBase = saturate(dot(reflect(-normalize(viewVec), normal),
lightDir));
// Perturb a grid pattern with some noise and with the view-vector
// to let the glittering change with view.
float3 fp = frac(0.7 * pos + 9 * Noise3D( pos * 0.04).r + 0.1 * viewVec);
fp *= (1 - fp);
float glitter = saturate(1 - 7 * (fp.x + fp.y + fp.z));
float sparkle = glitter * pow(specBase, 1.5);
```

Snow with Sparkle

AMD
Smarter Choice

Procedural Geometry

AMD
Smarter Choice

DirectX10 has added Stream Out to the programmer's arsenal

Geometry can now be assembled on the GPU

- Create new geometry on demand
- No need to transfer geometry data to the GPU

Icicles

AMD
Smarter Choice

Procedurally grow icicles using the GPU

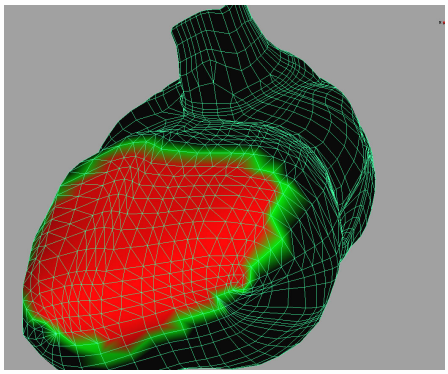
- From artist-specified locations
- According to artist-specified parameters

Specify locations with vertex color

Red = Triangles to be displaced

Green = Border case

(don't displace)



Considerations

AMD
Smarter Choice

Simply displacing vertices in the VS will not be enough

- Limited by resolution of the mesh
- Icicles will likely appear sharp and non-organic

Solution: Emit new geometry using stream out

- Add detail as necessary (subdivision)
- Perturb vertices downward according to icicle map and radially with noise

Emitting new geometry

AMD Smarter Choice

Pass 1:

- Emit all geometry with RED and GREEN vertex color into a dynamic vertex buffer
- Subtract value in icicle height map from y component of vertex position

Pass 2-N:

- For each triangle, if an edge is longer than some threshold *t*, subdivide that triangle (more on this->)
- Perturb new vertices by values from icicle height map

N = # passes before no new triangles are emitted

N can be determined by issuing a stream out statistics query (D3D10_QUERY_DATA_SO_STATISTICS) each pass

Subdivision

AMD
Smarter Choice

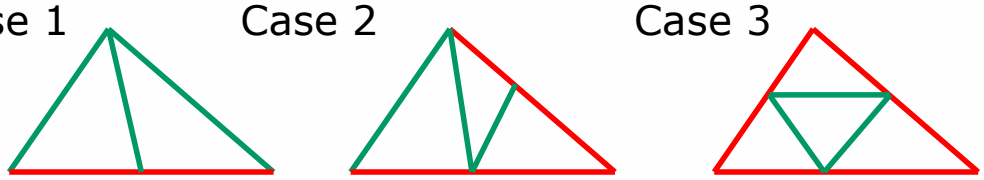
Can’t uniformly subdivide

Will introduce T-junctions and cracks

Only add new vertices on the edges that are too long

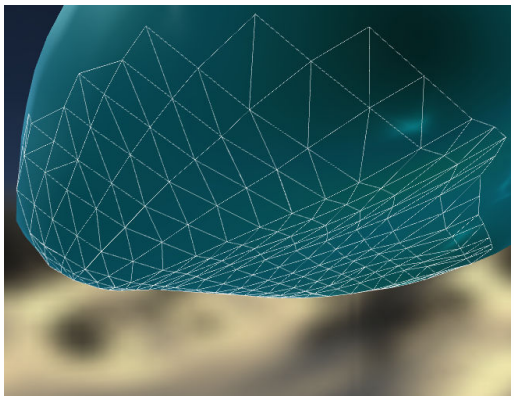
Have to handle three cases separately: 1, 2, and 3 long edges

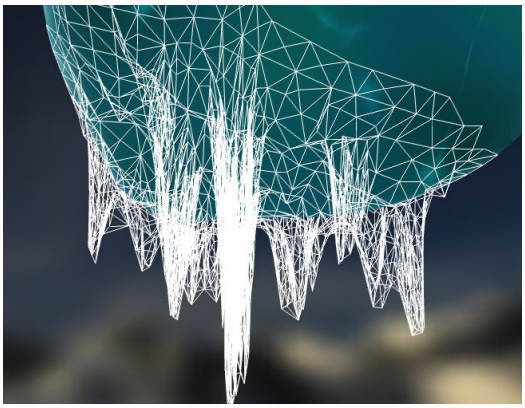
Case 1 Case 2 Case 3



Results of Subdivision

AMD
Smarter Choice





Authoring an icicle map

AMD
Smarter Choice

For this demonstration, we have used a simple noise texture for the icicle map

Better solution: generate it using simulation

Initialize simulation height map with depth values of surface from which icicles will hang (+ small noise)

Start:

- Introduce water drops at random positions, life = n
- Each frame, move water drops in the direction of greatest negative gradient in icicle map
- life--
- When no negative gradient is available or life == 0, add to the height map
- Goto Start

Lighting and Texturing

AMD
Smarter Choice

We don’t have normals for emitted vertices

- We could calculate per-face normals easily
- Doesn’t look good

Use the gradient of the icicle map we are using to drive the geometry generation

Use 3D position for indexing 3D noise for texturing

- No UVs for emitted geometry

Demo

Thank you

Natalya Tatarchuk

Dan Roeger

Daniel Szecket

Chris Oat

Thorsten Scheuermann

Josh Barczak